Estimating Functional Size of Software with Confidence Intervals

Luigi Lavazza Angela Locoro
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
email:luigi.lavazza, angela.locoro@uninsubria.it

Roberto Meli
DPO
Rome, Italy
email:roberto.meli@dpo.it

***Abstract*—In many projects, software functional size is measured via the IFPUG (International Function Point Users Group) Function Point Analysis method. However, applying Function Point Analysis using the IFPUG process is possible only when functional user requirements are known completely and in detail. To solve this problem, several early estimation methods have been proposed and have become *de facto* standard processes. Among these, a prominent one is the 'NESMA (Netherlands Software Metrics Association) estimated' (also known as High-level Function Point Analysis) method. The NESMA estimated method simplifies the measurement by assigning fixed weights to Base Functional Components, instead of determining the weights via the detailed analysis of data and transactions. This makes the process faster and cheaper, and applicable when some details concerning data and transactions are not yet known. The accuracy of the mentioned method has been evaluated, also via large-scale empirical studies, showing that the yielded approximate measures are sufficiently accurate for practical usage. However, a limitation of the method is that it provides a specific size estimate, while other methods can provide confidence intervals, i.e., they indicate with a given confidence level that the size to be estimated is in a range. In this paper, we aim to enhance the NESMA estimated method with the possibility of computing a confidence interval. To this end, we carry out an empirical study, using data from real-life projects. The proposed approach appears effective. We expect that the possibility to estimate that the size of an application is in a range will help project managers deal with the risks connected with inevitable estimation errors.**

***Index Terms*—Function Point Analysis; Early Size Estimation; High-Level FPA; NESMA estimated.**

I. Introduction

In the late seventies, Allan Albrecht introduced Function Points Analysis (FPA) at IBM [1], as a means to measure the functional size of software, with special reference to the "functional content" delivered by software providers. Albrecht aimed at defining a measure that might be correlated to the value of software from the perspective of a user, and could also be useful to assess the cost of developing software applications, based on functional user requirements.

FPA is a Functional Size Measurement Method (FSMM), compliant with the ISO/IEC 14143 standard, for measuring the size of a software application in the early stages of a project, generally before actual development starts. Accordingly, software size measures expressed in Function Points (FP) are often used for cost estimation.

The International Function Points User Group (IFPUG) is an association that keeps FPA up to date, publishes the official FP counting manual [2], and certifies professional FP counters. Unfortunately, in some conditions, performing the standard IFPUG measurement process may be too long and expensive, with respect to management needs, because standard FP measurement can be performed only when relatively complete and detailed requirements specifications are available, while functional measures could be needed much earlier for management purposes.

To tackle this problem, the IFPUG proposes Simple Function Points (SFP). This is an alternative way of measuring the functional size of software: while the SFP method is based on the same concepts as FPA, it requires less detailed information than FPA, so that it is applicable before complete and detailed requirements specifications are available; besides, if is faster and cheaper to apply. As such, it is often presented as a lightweight functional measurement method, also suitable for agile processes. Although the SFP method provides measures that are quantitatively similar to those yielded by FPA, it is not an approximation method for FPA; instead, it is a different measurement method that yields different measures.

Before SFP was proposed, many methods were invented and used to provide *estimates* of functional size measures, based on fewer or coarser-grained information than required by standard FPA. These methods are applied very early in software projects, even before deciding what process (e.g., agile or waterfall) will be used. Among these methods, one of the most widely used is the "NESMA estimated" method [3], which was developed by NESMA [4]. Using this method for size estimation was then suggested by IFPUG [5], which renamed the method High-Level FPA (HLFPA).

HLFPA has been evaluated by several studies, which found that the method is usable in practice to approximate traditional FPA values, since it yields reasonably accurate estimates, although it has been observed that the NESMA method tends to underestimate size, which is potentially dangerous.

Many estimation methods provide a "confidence interval", meaning that instead of providing a single value, they predict that the size is in an interval. The greater the required confidence, the greater the interval. Knowing the confidence interval is considered very useful by project managers, because

 ISBN: ISBNFILL

it helps managing the risk deriving from inevitable estimation errors and the inherent uncertainty of estimates. Unfortunately, the NESMA estimated method does not provide a confidence interval. This papers aim to enhancing the NESMA estimated method by equipping it with a mechanism to create a confidence interval.

The remainder of the paper is organized as follows. Section II provides an overview of FPA and the NESMA method. Section III describes the empirical study and its results, which are discussed in Section IV. In Section V, we discuss the threats to the validity of the study. Section VI reports about related work. Finally, in Section VII, we draw some conclusions and outline future work.

II. Background

Function Point Analysis was originally introduced by Albrecht to measure the size of data-processing systems from the point of view of end-users, with the goal of the estimating the value of an application and the development effort [1]. The critical fortunes of this measure led to the creation of the IFPUG (International Function Points User Group), which maintains the method and certifies professional measurers.

The "amount of functionality" released to the user can be evaluated by taking into account 1) the data used by the application to provide the required functions, and 2) the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Both data and transactions are counted on the basis of Functional User Requirements (FURs) specifications, and constitute the IFPUG Function Points measure.

FURs are modeled as a set of Base Functional Components (BFCs), which are the measurable elements of FURs: each of the identified BFCs is measured, and the size of the application is obtained as the sum of the sizes of BFCs. IFPUG BFCs are: data functions (also known as logical files), which are classified into Internal Logical Files (ILF) and External Interface Files (EIF); and Elementary Processes (EP)—also known as transaction functions—which are classified into External Inputs (EI), External Outputs (EO), and External inQuiries (EQ), according to the activities carried out within the considered process and the primary intent.

The complexity of a data function (ILF or EIF) depends on the RETs (Record Element Types), which indicate how many types of variations (e.g., sub-classes, in object-oriented terms) exist per logical data file, and DETs (Data Element Types), which indicate how many types of elementary information (e.g., attributes, in object-oriented terms) are contained in the given logical data file.

The complexity of a transaction depends on the number of FTRs—i.e., the number of File Types Referenced while performing the required operation—and the number of DETs—i.e., the number of types of elementary data—that the considered transaction sends and receives across the boundaries of the application. Details concerning the determination of complexity can be found in the official documentation [2].

The core of FPA involves three main activities:

1) Identifying data and transaction functions.
2) Classifying data functions as ILF or EIF and transactions as EI, EO or EQ.
3) Determining the complexity of each data or transaction function.

The first two of these activities can be carried out even if the FURs have not yet been fully detailed. On the contrary, activity 3 requires that all details are available, so that FP measurers can determine the number of RET or FTR and DET involved in every function. Activity 3 is relatively time- and effort-consuming [6].

HLFPA does not require activity 3, thus allowing for size estimation when FURs are not fully detailed: it only requires that the complete sets of data and transaction functions are identified and classified.

The SFP method [7] does not require activities 2 and 3: it only requires that the complete sets of data and transaction functions are identified.

Both the HLFPA and SFP methods let measurers skip the most time- and effort-consuming activity, thus both are relatively fast and cheap. The SFP method does not even require classification, making size estimation even faster and less subjective (since different measurers can sometimes classify differently the same transaction, based on the subjective perception of the transaction's primary intent).

NESMA defined two size estimation methods: the 'NESMA Indicative' and the 'NESMA Estimated' methods. IFPUG acknowledged these methods as early function point analysis methods, under the names of 'Indicative FPA' and 'High-Level FPA,' respectively [5]. The NESMA Indicative method proved definitely less accurate [8], [9]. Hence, in this paper, we consider only the NESMA Estimated method.

The NESMA Estimated method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of functions: ILF and EIF are assumed to be of low complexity, while EI, EQ and EO are assumed to be of average complexity. Hence, estimated size is computed as follows:

$$EstSize_{UFP} = 7\ \#ILF + 5\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ$$

where *#ILF* is the number of data functions of type ILF, *#EI* is the number of transaction functions of type EI, etc.

III. Empirical study

In this section, the empirical study is described: Section III-A described the dataset used for the reported analysis; Section III-B illustrates some considerations concerning the accuracy of the NESMA method that affect the study; Section III-C describes how the study was performed; finally, SectionIII-D describes the obtained results.

A. The dataset

In the empirical study, we use an ISBSG dataset [10], which has been extensively used for studies concerning functional size [11]–[16].

 ISBN: ISBNFILL

The ISBSG dataset contains several small project data. As a matter of fact, estimating the size of small projects is not very interesting. Based on these considerations, we removed from the dataset the projects smaller than 100 UFP (Unadjusted Function Points). The resulting dataset includes data from 140 projects having size in the [103, 4202] range. Some descriptive statistics for this dataset are given in Table I.

TABLE I
DESCRIPTIVE STATISTICS FOR THE ISBSG DATASET.

	UFP	*#ILF*	*#EIF*	*#EI*	*#EO*	*#EQ*	NESMA
Mean	801	22	20	35	37	37	730
Std	818	21	22	37	65	48	721
Median	475.5	14	14.5	22	10	20.5	463
Min	103	0	0	0	0	0	71
Max	4202	100	172	204	442	366	3755

B. The accuracy of the NESMA estimated method

As already observed in previous papers [16], [17], the NESMA estimated method tends to underestimate. Figure 1 shows that more than 75% of the estimates by NESMA have positive error. Being the error defined as the actual size (i.e., the size measured via the ISBSG standard FPA process) minus the estimate, positive error indicate underestimation.

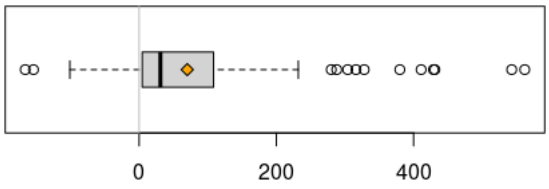

Fig. 1. Histogram of estimation errors by the NESMA method, when applied to the ISBSG dataset.

In addition, Figure 1 suggests that the distribution of NESMA errors is skewed. The skewedness of NESMA errors is clearly visible in Figure 2, which illustrates the distribution of errors: it is easy to notice that most errors are positive.

For our purposes, the fact that the distribution of NESMA errors is skewed and not centered on zero means that we cannot evaluate confidence errors as is usually done. Specifically, given a confidence level C we cannot select two error levels e_L and e_H that are symmetric with respect to the mean error $\bar{e}$ (i.e., $|e_H - \bar{e}| = |\bar{e} - e_L|$) such that the proportion of errors such that $e_H \geq error \geq e_L$ is C.

Since it makes hardly sense to provide confidence intervals for a method that underestimates systematically, we first "correct" the NESMA estimated method. Via a trial-and-error procedure, we found that by multiplying NESMA estimates by 1.08 it is possible to obtain estimates that have a better error distribution (less skewed and centered around zero) and a smaller mean absolute error (50.7 UFP instead of 83.8 UFP).

The boxplot of estimation errors obtained with the corrected NESMA method is shown in Figure 3. The error distribution is shown in Figure 4: it can be noticed that the distribution is much less skewed than in Figure 2.

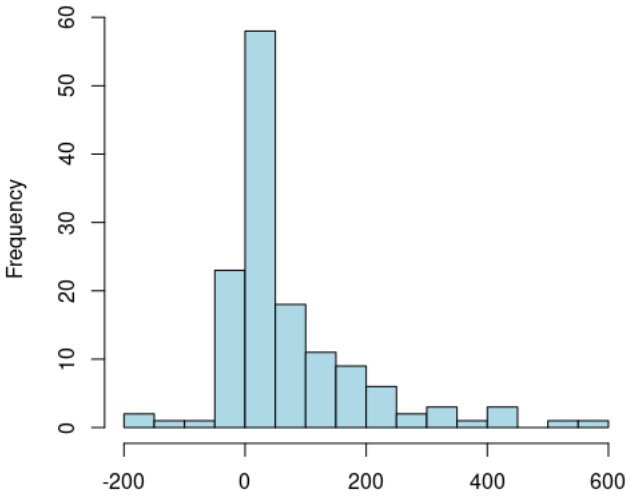

Fig. 2. Histogram of estimation errors by the NESMA method, when applied to the ISBSG dataset.

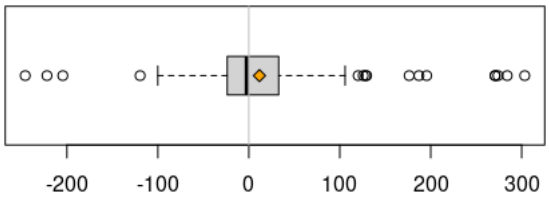

Fig. 3. Boxplot of estimation errors by the corrected NESMA method, when applied to the ISBSG dataset.

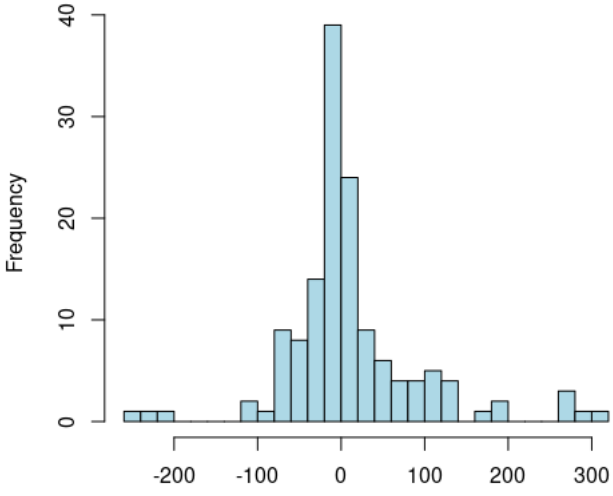

Fig. 4. Boxplot of estimation errors by the corrected NESMA method, when applied to the ISBSG dataset.

 ISBN: ISBNFILL

Since the practical objective of this work is to provide project managers with reliable predictions of functional size, in what follows we consider only estimates provided by the original NESMA method and corrected as described above. In other words, we consider the following estimates:

$$EstSize_{UFP} = 1.08\ (7\ \#ILF + 5\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ)$$

We make reference to this estimation as the "Corrected NESMA" method.

C. Method used

In essence, given a confidence level C we aim at finding two values k_L and k_H such that a proportion C of the actual size measures (i.e., measures obtained via the official IFPUG FPA process) is in the range $[k_L \cdot EstSize_{UFP}, k_H \cdot EstSize_{UFP}]$, where $EstSize_{UFP}$ is the size estimates computed via the Corrected NESMA method.

Finding k_L and k_H would be straightforward if the estimation errors obtained via the Corrected NESMA method were normally distributed. Instead, it is not so, as shown by the Shapiro-Wilk test.

Therefore, we proceeded as follows:

1) We computed the ratio $\frac{ActualSize}{EstSize_{UFP}}$ for all projects in the dataset, obtaining a set of ratios; this set was then sorted and stored in vector *vRatios*.
2) We computed the quantiles from 0 to 1, with 0.01 steps, of *vRatios*, obtaining an ordered vector *vQuant*.
3) We looked for two indexes i_L and i_H in *vQuant* such that $i_H - i_L + 1 = C \cdot n$ (where *n* is the number of projects in the dataset).
4) k_L and k_H are the values in *vRatios* having index i_L and i_H, respectively, i.e., $vRatios[i_L]$ and $vRatios[i_H]$.

In this way, we obtain a size estimate interval that contains a proportion C of all estimates, such that all estimates outside the interval are greater than those within the interval.

D. Results obtained

We applied the procedure described in Section III-C for various confidence levels. The results obtained are given in Table II. Note that these results depend on the dataset being used, in our case, the ISBSG dataset. In other contexts, a given confidence level could correspond to different confidence intervals. For instance, in the ISBSG dataset, the minimum and maximum ratios $\frac{ActualSize}{EstSize_{UFP}}$ are 0.758 and 1.343, respectively; in another dataset, a smaller minimum and a larger maximum ratios are clearly possible.

TABLE II
CONFIDENCE INTERVALS FOR VARIOUS CONFIDENCE LEVELS.

conf. level	k_L	k_H
0.50	0.947	1.053
0.60	0.933	1.069
0.70	0.902	1.100
0.80	0.875	1.134
0.90	0.840	1.165
0.95	0.826	1.220
1.00	0.758	1.343

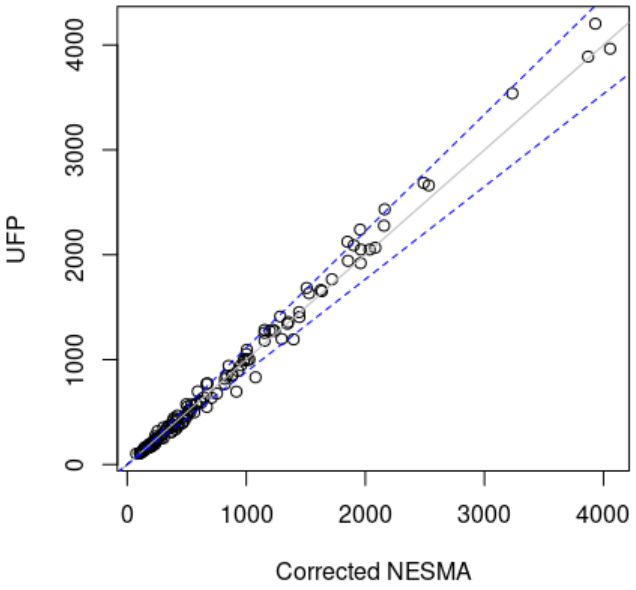

Fig. 5. Corrected NESMA estimates vs. actual size in UFP, with confidence $C = 0.75$.

For illustration purposes, Figure 5 plots the ISBSG project data in the plan defined by actual size (the y axis) and the size estimated via the Corrected NESMA method (the x axis). In the plot, the dashed blue lines represent the $y = k_L\ x$ and $y = k_H\ x$ lines.

IV. DISCUSSION OF RESULTS

In the previous sections, we exploited a dataset that collects measures from real-life projects to determine a) a correction of the estimates provides by the NESMA method, and b) confidence intervals for the corrected estimates.

The contribution of this paper is twofold:

– Organizations that own historical data like those we used can apply the procedure illustrated in Sections III-B and III-C to derive the correction constant and the confidence intervals that suite best their development process.
– Organizations that do not own historical data like those we used can use our correction constant (1.08) and the intervals in Table II, to get an idea of how much estimates obtained via the NESMA method can vary in practice. However, these organizations should be aware that the data we used might not match their situations, hence both the correction constant and the confidence intervals might not be perfectly suited for their case.

The confidence interval can be used to perform risk analysis. For instance, Table II shows that, given an estimate already corrected with respect to the NESMA original prediction, there are 30% probabilities that the actual size is more than 10% different (greater or smaller) than estimated. Most likely, half of these 30% probabilities concern the underestimation case: as a result, a project manager should consider that the probability of underestimating functional size of 10% or more is around 15%. The risk concerning the underestimation of

 ISBN: ISBNFILL

cost can be then computed, if the relationship between size and cost is known. So, the proposed method supports typical project management activities, like controlling the risk of size underestimation, that are not supported by the original NESMA size estimation method.

Finally, being the estimates obtained via the Corrected NESMA method proportional to the estimates obtained via the original NESMA method, the confidence intervals for the Corrected NESMA method can be easily converted into confidence intervals for the original NESMA method.

V. Threats to validity

The proposed approach is mostly empirical. From a theoretical point of view, the adopted practices may not be perfect, but the context itself suggests that this is not very relevant. The definition of the NESMA estimated method itself has no theoretically strong basis: it is simply the hypothesis—not experimentally verified—that on average data have low complexity (in FPA terms) while transactions have mid complexity. So, we looked for reasonable confidence intervals, although these intervals are not statistically linked to confidence levels in a rigorous way.

Another typical concern in this kind of studies is the generalizability of results outside the scope and context of the analyzed dataset. In our case, the ISBSG dataset is deemed the standard benchmark among the community, and it includes data from several application domains. Therefore, our results may be representative of a fairly comprehensive situation. However, additional studies are needed for confirming the general validity of this study. In the meanwhile, readers are reminded that the amount by which the NESMA method underestimates depend on the considered dataset; similarly, the confidence interval depends on the dataset. In both cases, the distribution of the complexity of BFCs (i.e., ILF, EIF, EI, EO and EQ) rules.

VI. Related Work

Measures for early software estimation were conceived since the last decades [18]–[20]. The present study aims to advance this field by providing statistical foundations to some of these measures, by using confidence intervals where approaches not based on probability distributions were adopted. For example, the "Early & Quick Function Point" (EQFP) method [21] estimates an error of $\pm 10\%$ of the real size of software, for most of the times, but fails to indicate a more robust indicator of this estimate, such as a confidence interval. Several other early estimation methods were proposed: Table III lists the most popular ones.

TABLE III
Early estimation methods: definitions and evaluations

Method name	Definition	Used functions	Weight	Evaluation
NESMA indicative	[22] [23]	data	fixed	[3] [17], [24]–[27] [9]
NESMA estimated	[22] [23]	all functions	fixed	[3] [17], [24]–[27] [9]
Early & Quick FP	[20] [28] [21]	all functions	statistics	[9] [29]
simplified FP (sFP)	[30]	all functions	fixed	[9]
ISBSG average weights	[31]	all functions	statistics	[9]
SiFP	[32]	data and trans.	statistics	[11] [13]

Recently, comparisons based on the accuracy of the HLFPA method and statistical modelling methods were carried out in order to assess whether standard measures fail in underestimating or overestimating software size [16].

A survey [33] reports how machine learning techniques were used for software development effort estimation, reporting accuracy as a comparison criterion for all the methods analysed. To the best of our knowledge, confidence intervals are overlooked as robust indicators of the estimates done in software size. In this respect, this study aims to emphasize the importance of providing robust indicators for a more reliable comparison and precision of reporting.

VII. Conclusion

The "NESMA estimated" method was proposed to estimate the functional size of software (expressed in IFPUG Function Points). The NESMA method assigns fixed weights to base functional components (i.e., ILF, EIF, EI, EO and EQ), so that it is not necessary to analyze in depth every logic data file or transaction. This makes the method both easier and faster, and applicable when the details needed to characterize and weight base functional components are not yet available.

Previous studies showed that the NESMA method is sufficiently accurate to be used in practice. However, it has two possibly relevant limitations: 1) it tends to underestimate the "real" (i.e., as obtained via the IFPUG FPA process) size of software, and 2) it yields a single estimate, with no confidence intervals. Both these characteristics can be be problematic for software project managers. In fact, planning a project based on underestimated size and, consequently, on underestimated effort estimates usually leads to unrealistic plans. Besides, getting a confidence interval for size estimates allows for evaluating the risks connected with imprecise size estimates.

In this paper, we have proposed a correction for the estimates yielded by the NESMA method, to avoid underestimation, and a procedure to compute the confidence interval. Both these contributions are expected to make project managers' life easier.

Acknowledgment

The work reported here was partly supported by Fondo per la Ricerca di Ateneo, Università degli Studi dell'Insubria.

References

[1] A. J. Albrecht, "Measuring application development productivity," in Proceedings of the joint SHARE/GUIDE/IBM application development symposium, vol. 10, 1979, pp. 83–92.
[2] International Function Point Users Group (IFPUG), "Function point counting practices manual, release 4.3.1," 2010.
[3] H. van Heeringen, E. van Gorp, and T. Prins, "Functional size measurement-accuracy versus costs–is it really worth it?" in Software Measurement European Forum (SMEF), 2009.
[4] nesma, "nesma site," https://nesma.org/ [retrieved: March, 2023].
[5] A. Timp, "uTip – Early Function Point Analysis and Consistent Cost Estimating," 2015, uTip # 03 – (version # 1.0 2015/07/01).
[6] L. Lavazza, "On the effort required by function point measurement phases," International Journal on Advances in Software, vol. 10, no. 1 & 2, 2017, pp. 108–120.
[7] IFPUG, "Simple Function Point (SFP) Counting Practices Manual Release 2.1," 2021.

 ISBN: ISBNFILL

[8] nesma, “Early Function Point Analysis,” https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/ [retrieved: March, 2023].
[9] L. Lavazza and G. Liu, “An empirical evaluation of simplified function point measurement processes,” Journal on Advances in Software, vol. 6, no. 1& 2, 2013, pp. 1–13.
[10] International Software Benchmarking Standards Group, ““Worldwide Software Development: The Benchmark, release 11,” ISBSG, 2009.
[11] L. Lavazza and R. Meli, “An evaluation of simple function point as a replacement of IFPUG function point,” in IWSM–MENSURA 2014. IEEE, 2014, pp. 196–206.
[12] L. Lavazza, S. Morasca, and D. Tosi, “An empirical study on the effect of programming languages on productivity,” in Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 1434–1439.
[13] F. Ferrucci, C. Gravino, and L. Lavazza, “Simple function points for effort estimation: a further assessment,” in 31st Annual ACM Symposium on Applied Computing. ACM, 2016, pp. 1428–1433.
[14] L. Lavazza, S. Morasca, and D. Tosi, “An empirical study on the factors affecting software development productivity,” E-Informatica Software Engineering Journal, vol. 12, no. 1, 2018, pp. 27–49.
[15] L. Lavazza, G. Liu, and R. Meli, “Productivity of software enhancement projects: an empirical study.” in IWSM-Mensura, 2020, pp. 1–15.
[16] G. Liu and L. Lavazza, “Early and quick function points analysis: Evaluations and proposals,” Journal of Systems and Software, vol. 174, 2021, p. 110888.
[17] L. Lavazza and G. Liu, “An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates,” in ICSEA, 2019, pp. 24–29.
[18] D. B. Bock and R. Klepper, “FP-S: a simplified function point counting method,” Journal of Systems and Software, vol. 18, no. 3, 1992, pp. 245–254.
[19] G. Horgan, S. Khaddaj, and P. Forte, “Construction of an FPA-type metric for early lifecycle estimation,” Information and Software Technology, vol. 40, no. 8, 1998, pp. 409–415.
[20] L. Santillo, M. Conte, and R. Meli, “Early & Quick Function Point: sizing more with less,” in 11th IEEE International Software Metrics Symposium (METRICS’05). IEEE, 2005, pp. 41–41.
[21] DPO, “Early & Quick Function Points Reference Manual - IFPUG version,” DPO, Roma, Italy, Tech. Rep. EQ&FP-IFPUG-31-RM-11-EN-P, April 2012.
[22] NESMA–the Netherlands Software Metrics Association, “Definitions and counting guidelines for the application of function point analysis. NESMA Functional Size Measurement method compliant to ISO/IEC 24570 version 2.1,” 2004.
[23] International Standards Organisation, “ISO/IEC 24570:2005 – Software Engineering – NESMA functional size measurement method version 2.1 – definitions and counting guidelines for the application of Function Point Analysis,” 2005.
[24] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, and N. Lester, “The value of software sizing,” Information and Software Technology, vol. 53, no. 11, 2011, pp. 1236–1249.
[25] J. Popović and D. Bojić, “A comparative evaluation of effort estimation methods in the software life cycle,” Computer Science and Information Systems, vol. 9, no. 1, 2012, pp. 455–484.
[26] P. Morrow, F. G. Wilkie, and I. McChesney, “Function point analysis using nesma: simplifying the sizing without simplifying the size,” Software Quality Journal, vol. 22, no. 4, 2014, pp. 611–660.
[27] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, “Assessing the effectiveness of approximate functional sizing approaches for effort estimation,” Information and Software Technology, vol. 123, July 2020.
[28] T. Iorio, R. Meli, and F. Perna, “Early&quick function points® v3. 0: enhancements for a publicly available method,” in SMEF, 2007, pp. 179–198.
[29] R. Meli, “Early & quick function point method-an empirical validation experiment,” in Int. Conf. on Advances and Trends in Software Engineering, Barcelona, Spain, 2015, pp. 14–22.
[30] L. Bernstein and C. M. Yuhas, Trustworthy systems through quantitative software engineering. John Wiley & Sons, 2005, vol. 1.
[31] R. Meli and L. Santillo, “Function point estimation methods: A comparative overview,” in FESMA, vol. 99. Citeseer, 1999, pp. 6–8.
[32] R. Meli, “Simple function point: a new functional size measurement method fully compliant with IFPUG 4.x,” in Software Measurement European Forum, 2011, pp. 145–152.
[33] M. N. Mahdi, M. H. Mohamed Zabil, A. R. Ahmad, R. Ismail, Y. Yusoff, L. K. Cheng, M. S. B. M. Azmi, H. Natiq, and H. Happala Naidu, “Software project management using machine learning technique—a review,” Applied Sciences, vol. 11, no. 11, 2021, p. 5183.

 ISBN: ISBNFILL